10/652,000 MN-892

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
17 November 2005 (17.11.2005)

PCT

(10) International Publication Number
WO 2005/109251 A2

(51) International Patent Classification[7]: G06F 17/30

(21) International Application Number:
PCT/US2005/015700

(22) International Filing Date: 4 May 2005 (04.05.2005)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
10/839,938 6 May 2004 (06.05.2004) US

(71) Applicant (*for all designated States except US*): ORACLE INTERNATIONAL CORPORATION [US/US]; 500 Oracle Parkway, M/S 5op7, Redwood Shores, CA 94065 (US).

(72) Inventor; and

(75) Inventor/Applicant (*for US only*): WU, Zhe [CN/US]; 40 Stillwater Drive, Nashua, NH 03062 (US).

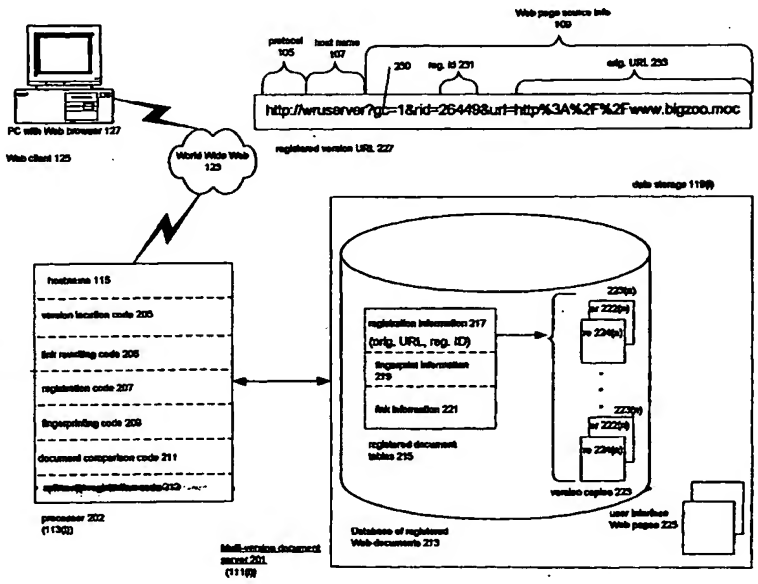
(74) Agent: NELSON, Gordon, E.; 57 Central Street, P.O. Box 782, Rowley, MA 01969 (US).

(81) Designated States (*unless otherwise indicated, for every kind of national protection available*): AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NA, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RU, SC, SD, SE, SG, SK, SL, SM, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, YU, ZA, ZM, ZW.

(84) Designated States (*unless otherwise indicated, for every kind of regional protection available*): ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IS, IT, LT, LU, MC, NL, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

[Continued on next page]

(54) Title: WEB SERVER FOR MULTI-VERSION WEB DOCUMENTS



(57) Abstract: A repository server that makes stored copies of Web-accessible documents available at times when the documents themselves are inaccessible via the Web, because the server the document is located on is unavailable, because the server has removed or renamed the document, or because the server has replaced the version of which the stored copy is a copy with a different version. A client of the repository server may register a document in the repository server. The repository server makes a copy of the registered document and returns a repository URL for the copy to the client. The repository URL may be used to fetch the copy from the repository URL. Registration further relates the stored copy to its document URL, to an identifier for the stored copy, to a fingerprint that is a condensed representation of the stored copy's content, and to a set of stored copies having similar content. Other operations performed by the repository server include: 1) fetching the content of a document that is registered in the repository server. This is done by receiving a repository URL and determining whether the document corresponding to the stored copy specified by the repository URL is available via the Web; if it is, the client is redirected to the location specified by the document's document URL; otherwise, the stored copy is fetched. 2) providing the client with a list of a repository URLs for stored copies that are related to the same document URL; and 3) providing the client with a list of repository URLs for stored copies that are similar to the stored copy specified by a given repository URL. The fingerprints are used to compute similarity.

WO 2005/109251 A2

Published:
— *without international search report and to be republished upon receipt of that report*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

Web server for multi-version Web documents

Background of the invention

1. Field of the invention

The invention is generally related to the Internet and more specifically related to the problem of accessing and tracking content that is accessible via the Internet.

2. Description of related art

The Internet, and in particular, the World Wide Web that is made possible by the Internet's HTTP protocol, have revolutionized the way in which we access information. FIG. 1 shows how the information access system 101 provided by World Wide Web 123 looks to a user of a computer 127 that has a Web browser and a hard drive 129 for persistent storage of data. Such a system is termed a web client 125. In addition to web clients 125, system 101 contains Web servers 111 that are accessible via world wide web 123. As shown in detail with regard to Web server 111(a), a Web server includes a processor 113(a) and data storage 119(a) which contains documents 121 which are accessible via the Web. These documents are termed in the following *Web documents*. A web document 121 may contain any kind or mixture of kinds of information; it may for example be an image or an audio file as well as a text document.

To access a document on the World Wide Web, a user of a Web browser in client 125 provides a *URL* (uniform resource locator) for the Web document to Web 123. Web 123 routes the URL to a web server 111(i) that contains the Web document specified by the URL. Web server 111(i) responds to the URL by providing the specified Web document via the Web to Web client 125. The browser then displays the Web document. Web documents typically contain *links*, i.e., URLs to other Web documents. When a user selects one of these links by clicking on it, the browser provides the URL to Web 123 and that Web document is provided to the web client by the Web server in which the Web document resides as just described.

An example URL is shown at 123. A URL has three main components: protocol 105, which specifies the Internet protocol that will be used to retrieve the Web document, in this case, the http protocol which is used in the World Wide Web, host name 107, which specifies Web server 111(i) upon which the Web document is stored, and Web page source info 109, which specifies how the Web document is to be located or otherwise produced in Web server 111(i). In example

URL 103, Web page source info 109 is a *pathname* which indicates how the Web document is to be located in a file system accessible to Web server 111(i); in other URLs, Web page source info 109 may specify a program that queries a database to locate the Web document or even a program that constructs all or part of the Web document on the fly. Web page source info 109 is interpreted in Web server 111(a) by executing source info interpretation code 117(a).

The complete syntax for a URL is the following:

```
<protocol_name>://<host_name>:<port_no>/<pathname>?
<parameter_list>
```

The `<protocol_name>`, `<host_name>`, and `<pathname>` have already been explained; `<port_no>` specifies the port on which Web server 111(a) is listening for the information specified by Web page source info 109; application programs for widely-used protocols such as the HTTP protocol have *default* port numbers which client 125 supplies for the protocol if no port number is specified in the URL. `<parameter_list>` is a list of parameters which are interpreted by source info interpretation code 117; the parameters may specify a program to be executed and data parameters for the program. The parameter list is made up of one or more parameter name-parameter value pairs that are separated by a & character:

```
<parameter_name>=<parameter_val>&...
  &<parameter_name>=<parameter_val>
```

Whenever a Web client 125 is connected to a physical network that provides access to World Wide Web 123, Web client 125 can access any Web server 111 that is operative at that time. Since most Web servers operate continually, most information that is available via the World Wide Web is available at any time from anywhere. Because that is so, Web users tend not to make copies of information that they have retrieved in Web client 125; instead, they save the URL of the Web document that contains the information in a list 131 of interesting URLs. One example of such a list is the "Favorites" or "Bookmarks" list provided by most Web browsers. When the user wants to access the information again, the user simply clicks on the URL in the Favorites list and thereby provides the URL to the browser.

Saving URLs instead of the Web documents they refer to has both advantages and disadvantages. Both stem from the dynamic nature of the World Wide Web. A URL is not a kind of library card catalog number for a Web document. A library card catalog number for a book uniquely identifies a particular edition of a book. If a new edition of the book comes out, it receives a new library card catalog number. The new card catalog number will be similar to the number for the other edition, since both editions will be classified in the same manner, but it

will not be identical to the number for the other edition. Because each edition has its own library card catalog number, a reader who writes down the card catalog number for a particular edition and ten years later presents the number to a library that has that edition will receive the edition.

A URL, by contrast, only identifies a Web server 111(i) and a Web document which the server will return in response to the Web page source info. There is no guarantee that the server specified by the URL will be available or even still exists, or that the Web document that the server will return is the same as the one that was there when the client saved the URL. What is actually returned is completely up to the server. The advantage of this arrangement is that what the server generally returns is the most recent version of the Web document. With many Web documents, for example, those which contain weather reports or stock market prices, that is exactly what is desired. The disadvantage is that older versions of the Web document are no longer accessible by the URL and may not be accessible at all. It is further generally not clear what relationship the currently-accessible Web document has to the older versions. One area where this causes difficulty is documentation for software. Increasingly, the manufacturer of the software provides such documentation by the World Wide Web; if the URL for the documentation specifies the current version of the software, a user who has an older version may be left with no documentation at all. About the only way the user of a Web browser 127 has to deal with this problem is to save a local copy of the documentation in his Web client. In so doing, of course, the user loses one of the most important advantages of the Web: the ability to save URLs instead of copies.

One attempt that has been made to deal with this problem is to establish Web archiving services such as the one found at `www.archive.org`. Such services have all of the problems of general-purpose archives: they are huge, but often do not have what the individual needs, and individuals typically have little or no input into what the archive saves. Additionally, vast amounts of the information which is accessible by a Web client is not publicly available and therefore will not be archived by an archiving service. This situation occurs when the Web server is behind a firewall which separates the public Internet from a so-called *intranet* which employs the Internet but is accessible only to Web clients known to the organization to which the intranet belongs. The server is thus accessible by Web clients that are also behind the firewall or that are known to the firewall, but not to Web clients in general. Such intranets are now one of the preferred ways of communicating within organizations.

It is an object of the invention disclosed herein to provide techniques for overcoming the foregoing problems of accessing documents by means of their URLs.

Summary of the invention

The object of the invention is attained by means of a repository server which fetches a document specified by a document URL, stores a copy of the document in the repository server, and provides a repository server URL for the stored copy which can be used by a Web client to fetch the stored copy. The repository server relates the stored copy to the document URL for the document and to an identifier in the repository server for the stored copy. The repository server may thus contain stored copies for several versions of a document represented by a particular document URL, with a unique repository server URL for each stored copy. The repository further creates a fingerprint for the stored copy. The fingerprint is a small encoded version of the stored copy which preserves information that characterizes both the structure and the content of the stored copy. Fingerprints are comparable to determine a degree of similarity of a pair of documents. Each stored copy's fingerprint is associated with the stored copy in the repository and the repository further associates a list of other stored copies that are similar to a given stored copy with the given stored copy.

A user interface consisting of pages provided to a client of the repository server permits a user of the client to *register* a document by specifying that operation and providing the document URL for the document. The repository server responds by making the stored copy, making the stored copy's repository URL, relating the stored copy to the document URL and the identifier for the stored copy, making the stored copy's fingerprint, and making the stored copy's list of similar stored copies.

The user interface further permits the user to fetch a document's content by specifying that operation and inputting the stored copy's repository URL to the repository server. The repository server responds to the repository URL by first using the document URL associated with the stored copy to determine whether the document is accessible in the network; if it is the client is redirected to the location specified by the document URL; if the document is not accessible, the stored copy specified by the repository URL is returned.

Additionally, the user interface permits the user to track stored copies of a document. The user specifies a repository URL and the operation in the client and the repository server responds by providing a list of repository URLs for stored copies that have the same document URL as the

document URL related to the copy specified by the repository URL. In one embodiment of this operation, the repository server also responds by providing a list of repository URLs for stored copies that are similar to the stored copy specified by the repository URL. Selection of a repository URL from either list causes the repository server to respond by providing the stored copy specified by the repository URL to the client.

In other aspects, the invention concerns techniques for making and comparing fingerprints, including techniques for determining whether two stored copies are similar enough to warrant a comparison of fingerprints, techniques for making the repository URL, and techniques for rewriting and using links in stored copies.

Other objects and advantages will be apparent to those skilled in the arts to which the invention pertains upon perusal of the following *Detailed Description* and drawing, wherein:

Brief description of the drawing

FIG. 1 shows a prior-art system for accessing information via the Internet;

FIG. 2 is a block diagram of the Web server of the invention;

FIG. 3 is the top-level Web page for the server's GUI;

FIG. 4 shows the Web page that appears if the page is not available on the Web;

FIG. 5 shows the Web page that appears if the user requests version tracking;

FIG. 6 is an entity-relationship drawing of registered document tables 215;

FIG. 7 shows an HTML document and a tree made from the document;

FIG. 8 shows an algorithm for making fingerprints;

FIG. 9 shows an algorithm for determining the similarity of documents by comparing the fingerprints from which they were made; and

FIG. 10 shows an algorithm for filtering documents for dissimilarity.

Reference numbers in the drawing have three or more digits: the two right-hand digits are reference numbers in the drawing indicated by the remaining digits. Thus, an item with the reference number *203* first appears as item 203 in FIG. 2.

Detailed Description

The following *Detailed Description* will first present an overview of the web server for multi-version documents, will then describe the user interface, and will finally present details of various components of the web server.

Overview of the Web server for multi-version documents: FIG. 2

FIG. 2 is a block diagram of a Web server 201 for multi-version Web documents. In some of the following, server 201 is termed a *repository server*. From the point of view of a Web client 125, Web server 201 is a standard Web server 111. When client 125 places a URL specifying Web server 201 on Web 123, Web 123 delivers the URL to server 201 and server 201 responds to the URL by returning a Web page to client 125. Like any other Web server, server 201 has two main components, a processor 202 which executes code in response to URLs specifying server 201's hostname 115, and storage for the Web pages specified by the URLs or for the information required to make the Web pages. Here, storage 119 includes web pages 225 for the user interface which Web client 125 uses to control document server 201 and a database of *registered Web documents* 213. A registered Web document is a Web document that has been *registered* in server 201, that is, server 201 has made a copy 223 of the Web document as it was when it was registered in database 213 and has made a registered version URL 227 to which multi-version document server 201 can respond by providing the version copy 223 referred to by registered version URL 227 to the Web client 125 that was the source of the registered version URL. In the preferred embodiment, copy 223 has two parts: an original copy 222, which is the copy exactly as it was received upon registration, and a rewritten copy 224, in which links have been rewritten. Why rewriting links is necessary and how it is done will be discussed in detail later.

Fetching a Web page with a registered version URL 227

Registered version URL 227 is a standard URL: it specifies a protocol 105, a host name 107, here, the host name of server 201, and Web page source info 109. The Web page source info in registered version URL 227 includes a parameter list which specifies a program 230 that is to be executed in processor 202 to provide the Web page and data to be used in the execution of the program. Here the program is gc, which fetches the content of a registered Web page. gc is one of the programs in version location code 203 executed by processor 202. When executing version location code 203, server 201 is an embodiment of what is later termed a *registered document provider*. gc takes two parameters: a registration identifier 231 which uniquely identifies the copy 223 of the registered Web page in database 213 and the original URL 233 of the fetched page. In a preferred embodiment, when multi-version document server 201 receives registered version URL 227, it executes the gc program. gc first attempts to establish a

connection between server 201 and the server 111(i) specified by original URL 233; if that succeeds, gc redirects Web client to fetch the Web page using original URL 233. If the attempt to establish a connection fails, gc uses registration ID 231 to locate copy 223 in database 213 and that copy is returned to Web client 125. When used as just described, multi-version document server 201 guarantees that when Web client 125 provides a registered version URL 227 to server 201, client 125 will receive the copy 223 of the Web page specified by registration identifier 231 even when the document specified by original URL 233 is unavailable.

The information that version location code 203 needs to interpret registered version URL 227 is contained in registration information 217 in registered document tables 215. Registration information 217 relates copy 223 to the original URL used to fetch the Web page from which copy 223 was made and to the registration identifier which uniquely identifies that copy 223. Because the registration identifier uniquely identifies copy 223, database 213 can contain different versions of the Web document identified by the original URL. Further, because the registration identifier is included in the registered version URL 227, registered version URLs are unique for each version and can be used by client 125 to fetch particular version copies 223 from server 201. In a preferred embodiment, this is done by inputting registration ID 231 from URL 227 into the user interface provided by server 201; in other versions, it may be done by providing URL 227 to the user interface and then specifying whether server 201 is to return the version currently specified by original URL 233 or the copy specified by registration identifier 231. As will be explained in more detail later, for a given copy 223(i), the user may also specify whether he or she wants original copy 222(i) or rewritten copy 224(i).

Registering a Web page in server 201

A user of a Web client registers a Web page in server 201 by inputting the URL of the Web page to be registered to a Web page of user interface web pages 225. In response to the input from that Web page, server 201 executes registration code 207 which fetches the document currently specified by the URL, generates a registration ID for it, creates an entry in registration information 217 for the registration ID which contains the original URL, the registration ID, and the location of a copy of the document in database 213, and then creates a URL 227 for copy 223 and returns it to the user, who can then use it as described above. When executing registration code 207, server 201 is an embodiment of what is later termed a *document registrar*.

Automated registration of Web pages

A preferred embodiment of server 201 will do automated registration of Web pages. In the preferred embodiment, an administrator of server 201 may specify a range of IP addresses of interest. Server 201 will execute automatic registration code 212 which fetches Web contents at those addresses and registers them as described above in server 201. Variations on automatic registration include:

1. Automatic registration of links: Starting from a registered URL, parse all the reference links in the document identified by the URL, and register those URL links automatically. The system can provide the option to limit registration of URLs only to those that come from the same website (domain).
2. Recursive automatic registration of links: Starting from a registered URL, recursively follow all of the URLs links in the document identified by the URL as long as they come from the same website (domain). Note that this option will be time-consuming for certain websites.
3. Automatic scheduled registration: setting up a schedule to register certain URLs of which content versions and changes are both frequent/periodic and important. For example, an URL pointing to an internal daily TODO list can be auto-registered using a daily schedule.

Of course, automatic registration of links and recursive automatic registration of links may be combined with other forms of automatic registration. Other embodiments of the invention may also provide a user interface which permits a user of server 201 to set up automatic registration of URLs that are of personal interest to the user from Web client 125. By using the document fingerprinting and fingerprint comparison techniques described below, server 201in anotherembodiment can limit automatic registration of documents to those which are substantially different from already registered documents.

Using server 201 to produce a version history of copies 223

Registration information 217 can also be used to provide a version history of the version copies 223 that were fetched using a particular original URL. Because registration information relates the original URL to the registration ID, processor 202 can query registration information 217 using a registration ID and receive a list of the registration identifiers for copies 223 of versions of the document that have the same original URL as the URL in for the copy 223 identified by the registration ID. Server 201 uses the registration IDs and the original URL to construct a list of URLs 227 for the copies 223 specified by the registration IDs. The list is returned as links in a Web page to Web client 125 and the user can click on a link to select a specific version. When

server 210 receives the URL 227 corresponding to the selected link, it uses the URL to return the copy 223 specified by the URL 227 to Web client 125. In other embodiments, server 201 may provide a version history in response to the input of an original URL or of a registered version URL 227. In the former case, the original URL would be used to make the query; in the latter case, original URL 233 from the registered version URL 227 would be used to make the query.

URL search

A server 201 may have many thousands of registered documents, each with its own copy 223 and registered version URL 227. In many embodiments, it will therefore be useful to provide a URL search capability. In URL search, the user at Web client 125 inputs a search string to a Web page 225 provided by server 201 and server 201 searches registration information 217 for original URLs which contain the search string input by the user. Wildcard characters may of course be used in the search strings. Server 201 can also offer case sensitive and insensitive search and sorting based on URLs or registration time. As output from the URL search operation, server 201 returns a Web page with a list of links containing matching URLs 227 and for each matching URL, its registration ID 231 and registration time.

It should be pointed out here that while server 201 registers URLs, the techniques just described can be used with any kind of locator or identifier that can be used to fetch a data item. For this reason, the generic term *document locator* is sometimes used in the following as a generic term for a URL of any other kind of locator or identifier that can be used to fetch a data item. What is required for registration is the following:

- Data structures available to server 201 that relate three things to each other:
  - a copy of the data item being registered that is accessible to server 201;
  - the document locator used to fetch the original of the data item at the time of registration; and
  - a registration identifier for the copy that uniquely identifies the copy that is accessible to server 201.
- A document locator for the copy to which server 201 responds by fetching the copy.

Operations performed on copies 223

Many operations can be performed on version copies 223 that further increase the usefulness of server 201. Among these are relative link rewriting, fingerprinting copies 223, comparing copies 223 on the basis of their fingerprints, and including indexes to the contents of the copies 223 in database 213.

Link rewriting

Web pages are defined using HTML (hypertext markup language). In HTML, a link to another Web page can be written either as an *absolute link*, which specifies a complete URL, including protocol 105, hostname 107, and Web page source info 109, or a *relative link*, which consists of Web page source info 109 which is appended to the URL used by the user to fetch the Web page that contains the relative link. Because the URL formed using the relative link always has the protocol and host name from the URL used to fetch the Web page containing the relative link, relative links are used to refer to other Web pages that are contained in the same host as the Web page that contains the relative link.

When a URL is registered in server 201 and a copy of the Web page specified by the URL is added to version copies 223, copy 223 is of course no longer in the Web server for the host for which the original of copy 223 was written and relative links in copy 223 will be completed from the URL used to fetch copy 223, which is a URL that specifies server 201. To deal with this problem, when server 201 fetches the Web page referred to by a URL which is being registered and adds a copy of the Web page to version copies 223, it makes two copies—one of these, original copy 222 is an exact copy of the original as fetched at registration time, the other, rewritten copy 224, is a copy in which all of the relative links in version copy 223 have been rewritten so that when the user clicks on a relative link, server 201 provides an absolute URL that like registered version URL 227 employs the `gc` program to redirect client 125 to the server 111(i) specified by an URL made from the hostname for the host from which the fetched Web page was obtained. The information server 201 needs to do this is obtained from the URL being registered. The preferred embodiment also rewrites the URLs for most absolute links in the same fashion, with the `gc program` redirecting client 125 to the server 111(i) specified in the original absolute link

Fingerprinting version copies 223

A *fingerprint* of a document is a value that is much smaller than the document and is produced from the document in such a fashion that the fingerprints of two documents can be compared to determine how similar the documents are to each other. Fingerprints differ from digests of

documents produced by hash functions. In that a comparison of two digests can only indicate whether the documents from which the digests were produced are identical. Fingerprint comparisons, on the other hand, indicate a degree of similarity of the documents being compared. Document server 201 employs fingerprinting code 209 to make a fingerprint of each registered document when the document is registered and the fingerprints are stored in fingerprint information 219. The fingerprinting algorithm used in a preferred embodiment will be explained in detail later.

Comparing version copies 223

Because each version copy 223 in server 201 has a fingerprint, the similarity between any two version copies 223(i) and (j) can be determined by comparing the fingerprint for version copy 223(i) with the fingerprint for version copy 223(j). The comparison is done by document comparison code 211. In a preferred embodiment, server 201 maintains for each version copy 223 a list of the results of fingerprint comparisons between the particular version copy 223 and version copies 223 whose original URLs have the same host name 107 as the particular version copy.

When the user of Web client 125 requests a version history for a given version copy 223, server 201 returns not only a list of links for the version copies 223 that have the same original URL 233 as the one associated with the registration ID 231 that the user provides as input to client 125, but also a list of links to version copies 223 that appear on the list of the results of the fingerprint comparisons. Included with the link is a value for the degree of similarity of the version copy specified by the link to the given version copy 223. In other embodiments, a user may be able to obtain a list of similar version copies independently of the version history operation simply by providing a registration ID for a document or selecting a link to the document from a list in a user interface Web page 225. Being able to show the degree of similarity between two version copies 223 has two important advantages:

- when a version history of versions referred to by a given original URL is made, the degree of similarity of the versions permits the user to make some surmises about the relationship between the versions; and
- Where the same or very similar content has been associated with two different original URLs from the same Web site, the degree of similarity between the copies 223 for the different original URLs reveals that the content associated with the original URLs is substantially the same.

Indexing version copies 223

The techniques described in USSN 10/810,756, Hu, et al., *A database management system with persistent, user-accessible bitmap values*, filed 3/26/04, may be employed to make bitmap indexes of the contents of version copies 223 in database 213. The indexes could be made for all of the version copies 223 or separate indexes could be made for the version copies 223 associated with a given original URL.

Examples of ways in which server 201 may be used

Server 201 has the following characteristics:

- it can provide a version copy 223 of a Web document when the Web site from which the document comes is unavailable.
- Registration makes a version copy 223 of the version of the Web document that is current available at the Web site from which the document comes and relates the copy and the original URL to a version number, so that versions of the Web document can be tracked.
- Registration may be automatic.
- Original URLs of version copies 223 may be searched.
- Version copies 223 are related to fingerprints 819 that permit comparison of the version copies.

Uses of server 201 which take advantage of these characteristics will be explained in the following.

Server 201 as a robust source for shared information

A business may have an intranet with a number of internal Web sites, including some personal Web pages located on PCs. A Web document on one of these sites is of course available only when the site is operating. A server 201 implemented in a high availability processor and database system can solve this problem. Web documents from the internal Web sites that need to be generally available at all times can be automatically registered in server 201. Workers for the business can use the registered version URL 227 for the document to fetch it. As described above, server 201 will respond to the registered version URL by first attempting to use the document's original URL 233 to redirect Web client 125 to the document's home Web site and if that fails, will provide the version copy 223 referred to by URL 227's registration ID 231. Server 201 used in this fashion may be combined with a search engine that has been modified to search the business's Intranet. When the search engine indexes a Web document, the document

is also registered with server 201, and the list of URLs returned by the search engine is a list of URLs 227 for registered copies 223 of the Web documents.

Server 201 as a source of versions of a Web document

Presently, if the business wishes to make historical versions of a Web document available through a single URL, the Web page produced in response to the single URL must be a list of the historical versions, and this list must be updated each time a new version becomes available. With server 201, the business need only register each new version in server 201; to see the available version copies 223 and retrieve any of them, a user need only use the version tracking operation with the registered version URL 227 for one of the versions.

Server 201 as a storage place for a version of a Web document that a user wishes to keep and make available to others

Presently, if someone finds a version of a Web document that is currently of interest to himself and others, those interested receive a URL for the Web document that may later not function at all or retrieve a version of the document that is different from the one that is currently of interest. This situation can be avoided by registering the version of the Web document that is of interest and providing its registered version URL 227 to those who are interested in the registered version.

Details of the user interface in a preferred embodiment: FIGs. 3-5

FIG. 3 shows the first user interface Web page 225 which server 201 returns to a user who wishes to use server 201 to register a Web page, fetch the Web page, or track the versions of a particular Web page that are stored in server 201. At 303 is a writable field into which a user may enter a URL to be registered, a registered version URL 227, or a registration ID number 231 to have server 201 fetch the original document or a copy 223 or track the versions of the document in server 201. When the URL or registration ID number has been input, the user clicks on one of the buttons under field 303 to specify the operation. As can be seen from the labels, when the user clicks on button 305, server 201 registers the URL in field 303. Upon registering the URL, server 301 returns a Web page with the contents shown at 315; as seen there, the returned page indicates the URL 317 that has been registered and the registered version URL 227 resulting from the registration. If the URL 317 has already been registered, message 315 so indicates.

When field 303 contains a URL 227 or a registration ID 231, what happens depends on which of buttons 307-313 the user clicks on.

- If the user clicks on `get contents` button 307, server 201 attempts to establish a connection to the server 111(i) specified by the document's original URL and redirect Web client 125 to that server. If the attempt fails, server 201 fetches the version copy 223 specified by the URL 227 or registration ID 231 and provides it to client 125. The Web page that appears when server 301 is able to redirect client 125 is simply the version that is currently available on server 111(i); if the redirection fails and server 301 fetches a version copy 223(i), what appears is Web page 401, shown in FIG. 4. Web page 401 is produced from rewritten copy 224 and includes highlighted legend 403 stating that links have been rewritten and indicating the date and time at which copy 223(i) was registered.
- If the user clicks on `Track it please` button 309, server 301 produces a Web page 501 (FIG. 5) which contains a list 503 of versions of copy 223 specified by the URL 227 or registration ID 231 and a list 505 of copies 223 that are similar to the copy specified by URL 227. In each list entry, the registration id of the copy 223 and the time the copy 223 was registered is specified; in the list of copies that are similar, each entry further contains a value for the degree of similarity at 507. At 509, finally, the page indicates when the tracking was done. Clicking on a list entry results in the return of the rewritten copy 224 of version copy 223 represented by the entry.
- If the user clicks on `Get that version` button 311, what is returned is rewritten copy 224 of the version copy 223 specified by the URL 227 or registration ID 231. The display is the same as that shown in FIG. 4.
- If the user clicks on `Get it in original` button 313, what is returned is original copy 222 of the version copy 223 specified by the URL 227 or registration ID 231.

Many other versions of this interface are of course possible. In some applications, not all of the operations will be required. For example, if server 201 is only intended to make sure that a version of a Web document is always available and documents are registered automatically or only by system managers, only a get contents operation may be needed. Similarly, if the only use of server 201 is to track versions and the versions are registered automatically or only by system managers, only the tracking operation may be needed In many embodiments, only system managers will be interested in original copy 224, so the user will only see rewritten copy 224. In other embodiments, the interface may provide buttons for other operations, for example,

a document comparison operation that is independent of the version tracking operation or a URL search operation.

Details of registered document tables 215

In a preferred embodiment, the information relating a registration ID 231 to the original URL 233 of a Web page as well as the information required for fingerprinting and document comparison is contained in registered document tables 215 in a relational database system. FIG. 6 is an entity-relationship diagram of tables 215 and the relationships between them. Dotted-line boxes in FIG. 6 indicate the relationship between the tables and the kinds of information shown in FIG. 2.

Beginning with registration information 217, this information is contained in two tables: PAGE_CACHE table 601 and REGISTERED_URL table 607. REGISTERED_URL table 607 relates registration IDs 231 to original URLs 233. There is a row in table 607 for each registered version copy 223. The fields in the row which are relevant to the present discussion are the REG_URL field 233, which contains the original URL 233 for the version copy 223, the REG_ID field, which contains the registration ID 231 for the version copy 223 and is the unique key for the row, and the REGISTER_TIME field, which contains a date stamp indicating the date and time at which version copy 223 was registered.

PAGE_CACHE table 601 relates registration IDs 231 to version copies 223. There is a row in table 601 for each registered version copy 223, and thus a row in table 601 for each row in REGISTERED_URL table 607. The fields in the row that are relevant to the present context are the REG_ID field, which contains the registration ID 231 for the version copy and is the unique key for the row, CONTENT_ENCODING, CONTENT_TYPE, and LAST_MODIFIED, which all contain information about the version copy 223, and the two versions of version copy 223: rewritten version 224 in the FULL_CONTENT field and original version 222 in the ORIG_CONTENT field. The occurrence of REG_ID in both the PAGE_CACHE and REGISTERED_URL tables further relates original URLs to version copies 223.

Fingerprint information 219 is contained in FINGER_PRINT table 609. There is a row in table 609 for each registration ID 231, and consequently for each version copy 223. The relevant fields in the row are the REG_ID field, which contains the registration ID 231 for the version copy 233 and is the unique key for the row, the FINGER_PRINT field, which contains the

fingerprint made from original version 222 of the version copy, and the SIMILAR_PAGES field, which contains a list of version copies 223 that come from the same Web site as the version copy 223 identified by the value of the REG_ID field and have a substantial similarity to the version copy identified by the value of the REG_ID field. The list includes for each version copy the version copy's registration ID and a value indicating the degree of similarity between that version copy and the version copy identified by the value of the row's REG_ID field. The LAST_FUZZY_MATCH_TIME field contains a time-date stamp that indicates the last time that the list in the SIMILAR_PAGES field was updated.

Link information 221 is contained in REFERENCED_LINK table 617 and ALL_LINK table 613. Each different link that is found in version copies 223 is identified by a unique link ID value. There is a row for each of the links in ALL_LINK table 613. The row for a link relates the link ID (in the LINK_ID field) to an absolute URL that corresponds to the URL specified by the absolute or relative link in the original Web page (in the LINK_URL field). The value of the LINK_ID field is the unique key for the row. REFERENCED_LINK table 617 relates registration IDs to the link IDs for the links in the version copy 223 specified by the registration ID. For each registration ID, there is a row for each link in the version copy 223 specified by the registration ID. The REG_ID field contains the registration ID 231 for the version copy and the LINK_ID field contains the link ID for one of the links in the REG_ID field. A query on REFERENCED_LINK table 617 by a registration ID will return a list of the link IDs for the links in the version copy 223 identified by the registration ID, and the link IDs can then be used in the ALL_LINK table to obtain the URLs for the links. In a preferred embodiment, link information 221 is used to make a threshold determination of whether two version copies 233 are similar enough to warrant comparing their fingerprints.

Details of the operations of Web server 201.

When the tables of FIG. 6 are studied in conjunction with the graphical user interface of FIGs. 3-5, it is immediately apparent how the operations of registering a URL, getting the contents of a registered URL, tracking versions, getting the rewritten version 224 of a copy 223, and getting the original version 222 of the copy 223 are performed.

Registration

When an original URL is entered in field 303 of entry user interface page 301 and the user clicks on Register URL button 305, registration code 207 obtains a unique registration ID 231 for

the registration, makes a row in PAGE_CACHE table 601 in which the REG_ID field's value is registration ID 231, and then saves a copy 222 of the content referred to by the original URL in the row's ORIG_CONTENT field. Registration code 207 further provides the relevant values from copy 222 for the CONTENT_ENCODING and CONTENT_TYPE fields and uses link rewriting code 205 to rewrite the links in copy 222 to produce copy 224, which is stored in the row's FULL_CONTENT field. Link rewriting code 205 also makes rows for the links as needed in ALL_LINK table 613 and REFERENCED_LINK table 617. Then the registration code makes a new row for the new registration identifier and the original URL in REGISTERED_URL table 607 and time stamps the new entry. Depending on the embodiment, registration code 207 may invoke fingerprinting code 209 fingerprinting of original version 222 of the new version copy 223 at registration time or fingerprinting may be done later. When it is done, a row is made in FINGER_PRINT table 609 for the new version copy 223 and the fingerprint. Since comparison of the fingerprint for the new version copy 223 with the fingerprints for all of the other version copies 223 from the same Web site requires considerable resources, making the list in SIMILAR_PAGES will typically be done later. Automatic registration works substantially as just described, except that the original URLs are provided by the Web crawler.

Getting contents

When a user inputs a registered version URL 227 or a registration ID 231 to field 303 and clicks on `Get contents` button 307, server 201 executes version location code 203. The input registration ID 231 or the registration ID 231 from the input registered version URL 227 is used to locate the row for the registration ID 231 in REGISTERED_URL table 607 and server 201 uses the original URL 233 in the REG_URL field to try to establish a connection with the server specified by original URL 233. If server 201 succeeds, it redirects Web client 125 to that server; otherwise, server 201 fetches rewritten copy 224 of the version copy identified by the registration ID from the registration ID's row in PAGE_CACHE 601. The date-time output in indication 403 that accompanies rewritten copy 224 is obtained from the REGISTER_TIME field in the registration ID's row in REGISTERED_URL table 607.

Tracking registered versions of a Web document having a particular original URL

When a user inputs a registered version URL 227 or a registration ID 231 to field 303 and clicks on `Track it please` button 309, server 201 also executes code that is part of version location code 203. Execution of this code causes server 201 to use the original URL associated

with the registration ID in registered version URL 227 or in REGISTERED_URL table 607 to locate all of the rows in REGISTERED_URL table 607 whose REG_URL field 233 has a value which is the same as that of the original URL. These rows of course correspond to the versions of Web documents stored in server 201 that correspond to the original URL. The code then causes server 201 to use these rows to produce a registered version URL 227 for each of the rows and to use the registered version URLs 227 to produce list of versions 503 in Web page 501.

Tracking registered versions of Web documents that are similar to a given registered version

In a preferred embodiment, Web page 501 also includes list 505 of registered Web documents 223 that are similar to the registered Web document 223 specified by registered version URL 227 or registration ID 231. In a preferred embodiment, version location code 203 causes server 201 to make list 505 from the list of registration IDs and similarity values in the field SIMILAR_PAGES in the row for the registration ID 231 in the table FINGER_PRINT 609.

Retrieving a particular version copy 223 of a Web document

When a user inputs a registered version URL 227 or a registration ID 231 to field 303 and clicks on Get that version button 311, version location code 203 causes server 201 to retrieve rewritten copy 224 from the field FULL_CONTENT in the row of PAGE_CACHE table 601 specified by the input registration ID 231 or the registration ID 231 from the input URL 227. When the user does the same and clicks on Get it in original button 313, version location code 203 causes server 201 to retrieve original copy 222 from the field ORIG_CONTENT in that row.

Details of link rewriting

As described above, when server 201 registers a Web document, server 201 rewrites the relative links in the Web document to produce rewritten copy 224 of the Web document. The code that server 201 executes to do the rewriting is link rewriting code 205, which employs the following method to rewrite each URL that it encounters in the Web document being registered:

URL rewriteUrl(URL baseURL, URL url).

{

1. First, construct an absoluteURL based on baseURL and url. Note that if url is not a relative URL, then absoluteURL will be equal to url.

2. Second, if the url is from HTML form action, for example <form action="...">, then return absoluteURL.
3. Otherwise, construct

   `http://wru-server-host-and-path?gc=1&url=absoluteURL` and return it. Note that what is constructed here is a special form of registered version URL 227 that includes original URL 233 but does not include registration ID 231.

Server 201 then writes this special form of URL 227 into rewritten copy 224 in place of the URL in the original link. When a user is viewing rewritten copy 224 and clicks on a link with a rewritten URL, server 201 receives the special form of registered version URL 227 and responds as described above for the standard registered version URL 227: it redirects Web client 125 to the Web document specified by the absolute URL contained in the special URL 227. In other embodiments, if that Web document is unavailable and there is a stored copy corresponding to the URL in server 201, server 201 may fetch the copy, and may in some cases fetch the copy whose registration date makes it most relevant to the copy that contains the link. Indeed, in some embodiments, server 201 may first attempt to fetch a relevant stored copy instead of fetching the current version. URLs in the HTML form action construct <form action="url"> are not rewritten using the special form of URL 227 because they are usually entry points to some processing logic that is available only in the Web server that was the source of the original document. An example is <form action="login.jsp">. To deal with this, the URLs in HTML form action are simply rewritten as absolute URLs for the source server.

Rewriting code 205 further causes server 201 to add a row to `ALL_LINK` table 613 whenever it encounters or makes an absolute URL that is not already in table 613 and to add a row in `REFERENCED_LINK` table 617 for version copy 223 to which rewritten copy 224 belongs whenever it encounters or makes an absolute URL for which there is not a row in table 617 for version copy 223's registration ID and the absolute URL's link ID in `ALL_LINK` table 613.

Details of fingerprinting

Overview

The whole flow for finding registered URLs for similar version copies 223 is as follows:

- Generate a fingerprint using the algorithm explained below for the copy 223 represented by a

- Place rows for the absolute links produced by the link rewriting done to produce rewritten version 224 as required in ALL_LINK table 613 and REFERENCED_LINK table 617.
- Given an original URL 233 for a given copy 223, get a list of the registration IDs 231 and fingerprints for all other copies 223 whose original URLs 233 have the same host name 107 - and port as the host name and port for the given copy's original URL.
- Filter the list using formal criteria to quickly eliminate copies 223 which have no meaningful similarity to the given copy 223. In the preferred embodiment, this is done using information in REFERENCED_LINK table 617.
- For the copies 223 that pass the filtering, apply the fuzzy matching algorithm defined below to the fingerprints.
- Make a list of the registration IDs 231 and degrees of similarity for the copies 223 that bear a reasonable degree of similarity to the given copy 223.

The fingerprinting algorithm: FIGs. 7 and 8

This section describes the technique used in a preferred embodiment to produce a fingerprint from an HTML document that can be compared with another fingerprint produced using the same technique to determine a degree of similarity between the HTML documents for which the fingerprints were made. The fingerprints can be compared to determine a degree of similarity because they contain reduced-size representations of all of the structural information contained in the HTML document along with reduced-size representations of the content of the HTML document. The technique described here for fingerprinting HTML documents can be adapted for use in any document representation in which the representation clearly separates the structural information about the document from the document's content.

As shown at 707 in FIG. 7, a HTML document can be viewed as a tree. In the tree, there are two kinds of nodes: nodes 709 representing the structural information in the HTML document and nodes 711 containing the actual content of the HTML document. The content nodes 711 generally contain text. The tree shown at 701 is made from the simple HTML document shown at 701. As shown there at 703, the structural information is provided by HTML tags. Each tag is enclosed in angle brackets, while content is not, as shown at 705. Tags generally come in pairs, for example, <html>, </html>, with the first tag of the pair indicating the beginning of the portion of the HTML document to which the tag applies and the second tag indicating the

end. A pair of tags may have other tags or pairs of tags nested in it. In tree 707, each tag node 709 contains the corresponding tag from document 701 and each text node 711 contains the corresponding text.

If fingerprints are to be useful for determining the similarity of HTML documents, a fingerprint must capture the structure of the document as defined by its HTML tags as well as its text or other content but must also be substantially smaller than the HTML document they are made from. In the technique used in the preferred embodiment, the size of the tags is reduced by encoding them and replacing the tags in the HTML document with their encoded versions. To reduce the size of the content in the fingerprint while retaining comparability of the content, several techniques are available:

1. Use a message digest algorithm like MD5 that generates a message digest of a fixed size from a message;
2. Hash the content using a standard hashing technique like the Secure Hash Standard;
3. Replace the text with a part thereof and perhaps also the length of the total text.

The first two techniques are well known. For details on MD5, see `http://www.faqs.org/rfcs/rfc1321.html`; for details on the Secure Hash Standard, see `http://www.itl.nist.gov/fipspubs/fip180-1.htm`. The preferred embodiment employs the third choice. For example, we can choose to use the first (or middle, or last) word plus the text length. We may also use the first (or middle, or last) non-blank character plus the text length. The third choice produces shorter representations of the text than the first two choices and requires less computation; its disadvantage is that it is less likely to detect changes in the text than the techniques that employ hash codes or digests.

In the preferred embodiment, the text is replaced by a characterization of the text made up of the text's first two non-blank lower case characters plus the text length info. Other embodiments may choose other characters, with the choice typically depending on the language. The length info is modularized by 58 and 64 is then added to the results of the modularization. The modularization and the addition of 64 serve to make the length information representable by human-readable ASCII characters. Note that 64 is the ASCII code for '@' and 121 is the ASCII code for 'y.' These two numbers, 64 and 58, are chosen heuristically and should be tunable system parameters. Other embodiments may use MD5 or some other hash standard for making the characterization of the text.

FIG. 8 shows the algorithm 805 and the fingerprint 819 resulting from the application of the algorithm to HTML document 701. At 801 are shown the encodings of the HTML tags that are employed in document 701. Each row of table 801 has the encodings for a start tag and its corresponding end tag. Thus, row 803 gives the encodings for <HTML> and </HTML>.

At 805 is shown the pseudocode for the fingerprinting algorithm employed in the preferred embodiment. At the top level, the algorithm has three parts: initialization 807, fingerprint production loop 809, and the return of the fingerprint at 817. Initialization 807 initializes the fingerprint variable to a null string. Loop 809 runs until the whole HTML document has been read and processed. There are three different kinds of nodes in tree 707: those for HTML start tags, those for HTML end tags, and those that contain content. `IF` statement 811 handles HTML start tags: it simply finds the encoded form corresponding to the start tag and places the encoded form in the fingerprint; `IF` statement 813 does the same with HTML end tags. Content nodes 711 are handled by `IF` statement 815, which works as described above to add the first two non-blank lower-case characters of the text and then a third character whose value is determined by the length of the text to the fingerprint.

Fingerprint 819 in FIG. 8 has been generated using the above algorithm from HTML document 701. The encoded <HTML> start tag is shown at 821, the encoded </HTML> end tag is shown at 825, and the string used to characterize the text `Hello!` is shown at 823. In terms of the tree of FIG. 707, the algorithm produces fingerprint 819 by processing node 709(b) and adding the result of the processing to fingerprint 819, and so on for node 709(d), node 709(f), node 711(a), node 709(g), node 709(h), node 711(b), node 709(i), node 709(e), and node 709(c).

Calculating similarities of HTML documents using fingerprints: FIG. 9

Intuitively, given two character-string fingerprints produced as just described from two HTML documents, the greater the total number of characters in the substrings the fingerprints have in common, the greater the degree of similarity of the HTML documents from which the two fingerprints were produced.

FIG. 9 is pseudocode for an algorithm 901 that is based on the above intuition. The details of the algorithm presented below depend on the representations of the HTML start tags used in the fingerprints, but the algorithm can easily be adapted to fingerprints made from other document representations. The algorithm takes as arguments a pair of fingerprints `fp1` and `fp2`, with

fp_ being no longer than fp2. The algorithm starts with the shorter fingerprint, termed the *source* fingerprint, locates the first '<' which is a delimiter of mapped HTML tags in the source fingerprint, finds the longest common substring that starts from the position located, then moves to the next '<' after the common substring and repeats this process. The algorithm uses two functions that need to be understood: `locate_substring()` and `longestSubstr()`.

`locate_SubString()` locates the next matching substring. The function takes three parameters. The first one is the source fingerprint. The second one is the substring that is to be found in the source fingerprint. The third parameter defines the starting position in the source string for the match. For example, `locateSubString("abcabc", "ab", 2)` will return 4. If no match is found, `locateSubString` will return 0.

`longestSubStr(fp1, fp2, delimPos)` returns the longest possible substring which starts from position `delimPos` in `fp1`, and is also a substring in `fp2`. For example, `longestSubStr("xyABCGGD1234", "88890ABCGGABZGGD1234", 3)` will return "ABCGG".

Continuing in more detail, at 903, an initialization step initializes two variables which mark positions in the source string, `lastPos`, which marks the current position in the source fingerprint from which matching is to begin, and `delimPos`, which marks the position of a current "<" delimiter character in the source fingerprint. `lastPos` is initialized to 1 and `delimPos` to the first "<" in the source fingerprint. The similarity between the fingerprints is initialized to 0. The processing of the two fingerprints is done in do while loop 905, which runs until no further left angle brackets "<"can be found in the source fingerprint. In the loop body, the first thing that is done is to look for the longest substring match between the part of `fp1` and beginning at the current "<" character and all of `fp2` (906); if the length of the match is greater than or equal to 6 characters, the length of the match is added to the variable `similarity`. The length of the current match or at least 1 is added to `lastPos`, and starting at `lastPos`, the next "<" character is found and loop 905 is repeated. When the loop terminates, the similarity of the documents is computed by dividing the current value of the variable `similarity` by the length of the source fingerprint.

Note that the number "6" in `IF` statement 906 is heuristically chosen. It should be part of the tunable system parameters. The purpose of this parameter is to put a lower bound on the size of

the common substring found. Intuitively, if only a match of a very short substring is found, for example "<v" which is "<p>" in HTML, the fact that such a match is found says little about the similarity of the HTML documents from which the fingerprints were made. As a possible improvement to the above algorithm, once a common substring has been located, it can be removed from `fp2` so that that part of `fp2` will not be further tested for matches. This is a bit more time consuming though.

Limiting the number of fingerprint comparisons to be made: FIG. 10

Since there is a fingerprint for every version copy 223 in server 201, the fingerprint 819 for any version copy can be compared with that for any other version copy 223. In most cases, of course, the comparison will show no similarity; finding that out from the fingerprints, however, requires the performance of algorithm 901, whose time of execution is slower for dissimilar fingerprints than for similar ones. What is needed here is a way of filtering document pairs prior to making a fingerprint comparison so that the fingerprint comparison is made only with documents that have a reasonable likelihood of being similar.

In a preferred embodiment, two separate techniques are used to filter document pairs. One technique is used in making the `SIMILAR_PAGES` list in a given page's row in `FINGER_PRINT` table 609. The pages on the list are limited to pages from the same Web site as the given page. Another technique is to check for dissimilarity of the links contained in the version copies 223 that are to be compared for similarity. This technique is based on the following observations:

- Similar HTML documents tend to have similar numbers of links and the links tend to point to the same Web sites in both documents.
- The links in an HTML document change at a rate which is much slower than the rate at which the HTML document's content or tags change.

From the foregoing, if it can be determined for a pair of version copies 223 that are being compared by document server 201 either that the two documents in the pair have greatly differing numbers of links or that the links do not point to the same Web sites in the two documents of the pair, it is clear that the copies 223 of the pair are not similar, and there is no need to compare their fingerprints. The two filtering techniques can of course be combined. It will of course be immediately apparent that the observations concerning links in HTML documents apply equally to links in any other document representation and that the filtering techniques can be used with any kind of document that contains links.

Server 201 maintains the information about links that it needs to filter on the basis of link dissimilarity in REFERENCED_LINK table 617. All the information that is needed for the filtering operation can be obtained by querying REFERENCED_LINK table 617 with the registration IDs of the copies 223 that are being compared. The query for each copy 223 of the pair determines the number of links in for each version copy 223 in a pair. The query further returns a list of the LINK_ID values for the links in the version copy 223 and these values can be compared. If the same LINK_ID value appears on both lists, both version copies 223 have a link to the same URL.

The algorithm for filtering on the basis of link dissimilarity is shown in FIG. 10. As would be expected from the foregoing discussion, the algorithm is carried out in two steps. The first step 1001 determines how many links each version copy of the pair contains and filters on that basis. linkNum1 contains the number of links in the first copy 223 in the pair and linkNum2 contains the number of links in the second copy 223 in the pair. Two tests 1003 and 1005 are applied and the pair of copies 223 is filtered out unless it passes one of them. Test 1003 simply lets pairs of copies 223 pass where the combined number of links is no greater than 7. Test 1005 divides the difference between the number of links in the two documents by the sum of the number of links +2 in the two documents and lets the pair pass if the result of the division is less than 1/3. Again, the numbers 7, 2 and 0.3333, used are chosen heuristically. They could be part of the system's tunable configuration parameters. The first condition says that if together these two URLs contain just a few reference links, the filter should let it pass. The second condition says that if the numbers of reference links of these two URLs differ within a predefined range, the filter should let it pass.

Second step 1007 determines the number of links that the two pages have in common. totalLinksNum is the number of links in one of the copies 223 of the pair; sameLinksNum is the number of identical URLs for which there are links in both of the copies 223. The condition for passing is shown at 1015. Again, the numbers used in the test are heuristically chosen and should be tunable system configuration parameters. The condition says that if the number of the same links is reasonably large compared to the total number of reference links, the filter should let the pair of copies 223 pass. When a pair of copies passes both test 1001 and test 1007, server 201 will compare the fingerprints for the pair.

The foregoing *Detailed Description* has described to those skilled in the relevant technologies how to make and use the repository server of the invention and has further described the best mode presently known to the inventor of implementing his repository server. It will be immediately apparent to those skilled in the relevant technologies that many implementations of the invention are possible other than the preferred embodiment described herein. For example, the preferred embodiment operates in the environment provided by the Internet and the `http` protocol and consequently, many details of the preferred embodiment are determined by that environment. The principles of the invention can, however, be employed in any situation where there is a need to ensure that copies of documents normally accessible via a network are accessible when they are not available via the network or when different versions are needed from those presently accessible via the network.

In the preferred embodiment, the necessary relationships between the copies stored in the repository, the document URLs for the copies, the identifiers that identify the copies in the repository server, the copies and their fingerprints, and a copy and its list of similar copies are maintained using tables in a relational database system. These tables may of course take many forms other than the forms disclosed herein, and beyond that, any arrangement of data structures that permits the necessary relationships to be represented will work as well as the relational tables. A given implementation of the invention may not perform all of the operations performed by the preferred embodiment or may perform operations that the preferred embodiment does not perform. The user interface for the preferred embodiment is of course determined by the requirements of the `http` protocol, by the operations performed by the preferred embodiment, and by the taste of the inventor, and many other user interfaces are possible.

With the fingerprints, too, many details of the preferred embodiment are a consequence of the fact that the fingerprints are designed to be made from HTML documents, but the techniques described herein for making and comparing fingerprints may be easily adapted to any form of document which has structural components as well as content.

For all of the foregoing reasons, the *Detailed Description* is to be regarded as being in all respects exemplary and not restrictive, and the breadth of the invention disclosed herein is to be determined not from the *Detailed Description*, but rather from the claims as interpreted with the full breadth permitted by the patent laws.

What is claimed is:

1. A repository server that is accessible via a network and provides documents in response to document locators received via the network from clients,
the repository server comprising:

a document registrar that receives a first document locator for a document and responds thereto by using the first document locator to fetch a copy of the document via the network from another server, storing the copy in storage accessible to the repository server, and making a second document locator for the copy; and

a registered document provider that responds when the second document locator is received in the repository server by providing the stored copy to the client.

2. The repository server set forth in claim 1 wherein:

the registered document provider indicates on the provided stored copy that a stored copy has been provided.

3. The repository server set forth in claim 1 wherein:

the document registrar associates the first document locator with the stored copy; and

the registered document provider further responds when the second document locator is received in the repository server by using the associated first document locator to determine whether the document is currently accessible via the network and providing the stored copy to the client only when the document is not currently accessible via the network.

4. The repository server set forth in claim 3 wherein:

the repository server receives an indication together with the second document locator that indicates whether the stored copy is to be provided; and

the registered document provider responds thereto by providing the stored copy to the client.

5. The repository server set forth in claim 3 wherein:

when the registered document provider determines that the document is currently accessible via the network, the registered document provider uses the associated first document locator to redirect the client to the other server.

6. The repository server set forth in claim 5 wherein:

ıent locator includes a copy of the first document locator; and

the registered document provider uses the copy of the first document locator as the associated first document locator.

7. The repository server set forth in claim 1 wherein:

the document registrar associates a registration identifier with the stored copy; and

the registered document provider responds when the registration identifier is received in the repository server by providing the stored copy to the client.

8. The repository server set forth in claim 1 wherein:

the document registrar rewrites relative links in the stored copy as absolute links.

9. The repository server set forth in claim 8 wherein:

each rewritten absolute link contains a document locator that specifies the repository server and includes a document locator that specifies the document specified by the relative link corresponding to the absolute link; and

the registered document provider responds to the rewritten absolute link by using the included document locator to determine whether the document specified by the rewritten absolute link is currently accessible via the network and if so, to redirect the client as specified in the included document locator.

10. The repository server set forth in claim 9 wherein:

when the document specified by the rewritten absolute link is not currently accessible via the network and there is a stored copy associated with the included document locator in the repository server, the registered document provider returns the associated stored copy.

11. The repository server set forth in claim 1 wherein:

the document registrar receives the first document locator from a client.

12. The repository server set forth in claim 1 wherein:

the document registrar receives the first document locator from a list thereof in the repository server.

13. The repository server set forth in claim 1 wherein:

at least some of the first document locators on the list are provided to the repository server by a network crawler.

14. A data storage device characterized in that:

the data storage device contains code which when executed in a processor implements the repository server set forth in claim 1.

15. A method of registering a document that is accessible by using a first document locator in a network in a repository server that is accessible via the network,
the method comprising the steps performed in the repository server of:

receiving the first document locator for the document in the repository server;

using the first document locator to fetch the document;

storing a copy of the document in storage accessible to the repository server; and

making a second document locator to which the repository server responds by providing the stored copy.

16. The method set forth in claim 15 further comprising the step of:

rewriting relative links in the stored copy as absolute links.

17. The method set forth in claim 16 wherein:

each rewritten absolute link includes a third document locator which includes the document locator for the document and to which the repository server responds by redirecting a client from which the third document locator was received as specified by the included document locator.

18. The method set forth in claim 15 further comprising the step of:

relating the stored copy to the first document locator.

19. The method set forth in claim 18 wherein:

the stored copy is related to the first document locator by including the first document locator in the second document locator.

20. The method set forth in claim 19 further comprising the step of:

relating the stored copy to a registration ID to which the repository server responds by providing the stored copy.

21. The method set forth in claim 19 wherein:

the stored copy is related to the registration ID by including the registration ID in the second document locator.

22. The method set forth in claim 15 wherein:

in the step of receiving, the first document locator is received from a client of the repository server.

23. The method set forth in claim 15 wherein:

in the step of receiving, the first document locator is received from a list thereof in the repository server.

24. The method set forth in claim 23 wherein:

at least some of the first document locators on the list are provided to the repository server by a network crawler.

25. A data storage device characterized in that:

the data storage device contains code which when executed in a processor implements the method of registering a document set forth in claim 15.

26. A method of providing a document for which a stored copy has been registered in a repository server that is accessible via a network and provides documents in response to document locators received via the network from clients,

the method comprising the steps performed in the repository server of:

receiving a document locator for the stored copy from a client;

responding thereto by determining whether the document can be fetched via the network; and

if the document cannot be fetched, providing the stored copy to the client.

27. The method set forth in claim 26 further comprising the step performed when the stored copy is provided to the client of:

indicating on the stored copy that the stored copy has been provided.

28. The method set forth in claim 26 wherein

the stored copy is related to the document locator for the document in the repository server; and

the method further comprises the step of:

when the document can be fetched via the network, redirecting the client as indicated by the related document locator for the document.

29. The method set forth in claim 28 wherein:

the related document locator for the document is contained in the document locator for the copy.

30. The method set forth in claim 26 wherein:

the stored copy is related to a registration identifier for the stored copy;

the document locator is the registration identifier; and

the step of providing the stored copy to the client is performed regardless of whether the document can be fetched via the network.

31. A data storage device characterized in that:

the data storage device contains code which when executed in a processor implements the method of providing a document set forth in claim 26.

32. A method of providing a document for which a stored copy has been registered in a repository server that is accessible via a network and provides documents in response to document locators received via the network from clients,

the method comprising the steps performed in the repository server of:

receiving a document locator for the stored copy and an indication whether the copy or the document is to be provided from a client;

responding thereto if the indication indicates that the stored copy is to be provided by providing the stored copy; and

responding otherwise by determining whether the document can be fetched via the network; and

if the document cannot be fetched, providing the stored copy to the client.

33. A data storage device characterized in that:

the data storage device contains code which when executed in a processor implements the method of providing a document set forth in claim 32.

34. A repository server that is accessible via a network and has access to stored copies of documents identified by first document locators,
the repository server comprising:

a document registrar that receives a first document locator for a document and responds thereto by using the first document locator to fetch a copy of the document via the network from another server, storing the copy in storage accessible to the repository server, relating the first document locator to the copy, and making an identifier for the copy; and

a registered document provider that responds when the identifier is received in the repository server from a client by providing a list of the identifiers to the client, the identifiers on the list indicating stored copies that are related to the same first document locator as the stored copy specified by the received identifier.

35. The repository server set forth in claim 34 wherein:

the document registrar further associates a time of registration with the stored copy; and

the registered document provider further provides the stored copy's time of registration for each of the stored copies specified by the identifiers on the list.

36. The repository server set forth in claim 34 wherein:

the registered document provider further responds to an identifier from the list that is received from the client by providing the stored copy specified thereby.

37. The repository server set forth in claim 34 wherein:

the identifiers on the list are second document locators specifying stored copies in the repository server.

38. The repository server set forth in claim 37 wherein:

the received identifier is a second document locator specifying a stored copy in the repository server.

39. A data storage device characterized in that:

the data storage device contains code which when executed in a processor implements the repository server set forth in claim 34.

40. A method of making versions of a document that are stored in a repository server that is accessible via a network accessible to a client of the repository server, the document being or having been formerly stored in another server that was accessible via the network and being or having been accessed in the other server by specifying a first document locator, each one of the stored versions being related to the first document locator and to an identifier in the repository server and

the method comprising the steps performed in the repository server of:

receiving the identifier for a given stored version from the client;

using the first document locator to which the given version is related to find other stored versions related to the same first document locator; and

providing a list of identifiers for the other stored versions to the client, whereby the client may access a stored version of the other stored versions.

41. The method set forth in claim 40 wherein:

the repository server further relates stored versions to the times at which the stored versions were stored in the repository server; and

the step of providing a list of identifiers includes providing the times at which the stored versions specified by the second document servers were stored.

42. The method set forth in claim 40 further comprising the step of:

responding to reception from the client of an identifier from the provided list by providing the stored version specified thereby to the client

43. The repository server set forth in claim #42wherein:

the identifiers on the provided list are second document locators specifying stored copies in the repository server.

44. The repository server set forth in claim #43 wherein;

the received identifier is a second document locator specifying a stored copy in the repository server.

45. A data storage device characterized in that:

the data storage device contains code which when executed in a processor implements the repository server set forth in claim 40.

46. A repository server that is accessible via a network and has access to stored copies of documents identified by first document locators,
the repository server comprising:

a document registrar that receives a first document locator for a document and responds thereto by using the first document locator to fetch a copy of the document via the network from another server, storing the copy in storage accessible to the repository server, computing similarities between the stored copy and other stored copies in the repository server, relating the stored copy to similar other stored copies, and making a first identifier for the copy; and

a registered document provider that responds when the first identifier is received in the repository server from a client by providing a list of second identifiers for the similar other stored copies that are related to the stored copy specified by the received identifier.

47. The repository server set forth in claim 46 wherein:

the document registrar further computes similarity values, each similarity value indicating a degree of similarity of the stored copy with one of the related similar other stored copies with the stored copy; and

the registered document provider further provides the other stored copy's similarity value with the other stored copy's second identifier.

48. The repository set forth in claim 47 wherein:

the document registrar further relates a similarity list to the stored copy, the list indicating for each similar other copy the similarity value for the similar other copy.

49. The repository server set forth in claim 46 wherein:

the document registrar computes the similarities between the stored copy and the other stored copies by computing fingerprints of the stored copy and the other stored copies and comparing the fingerprint of the stored copy with the fingerprints of the other stored copies.

50. The repository server set forth in claim 49 wherein:

the registrar relates a stored copy's fingerprint to the stored copy.

51. The repository server set forth in claim 46 wherein:

the registered document provider further responds to an identifier from the list that is received from the client by providing the stored copy specified thereby.

52. The repository server set forth in claim 51 wherein:

the second identifiers are second document locators specifying stored copies in the repository server.

53. The repository server set forth in claim 52 wherein;

the first identifier is a second document locator specifying a stored copy in the repository server.

54. A data storage device characterized in that:

the data storage device contains code which when executed in a processor implements the repository server set forth in claim 46.

55. A method performed in a repository server that is accessible via a network and that contains stored copies of documents that are obtained from other servers and are identified in the repository server by identifiers of making other stored copies that are similar to a given stored copy available to a client of the repository server,

the method comprising the steps of:

computing similarities between the given stored copy and other stored copies;

relating the ~~stored copy to similar other stored copies;~~

responding to an identifier for the given stored copy received from the client by providing a list of the identifiers to the similar other stored copies; and

responding to an identifier ~~which has~~ been selected in the client from the provided list by providing the other stored copy specified by the identifier to the client.

56. The method set forth in claim 55 wherein:

the step of computing similarities further comprises the step of computing a similarity value for each similar other stored copy, the similarity value indicating a degree of similarity of the given stored copy with the similar other stored copy;

the step of relating the given stored copy to similar other stored copies further comprises the step of relating the similarity values to the given stored copy; and

the step of responding to an identifier for the given stored copy further comprises the step of providing the similarity values for the similar other stored copies.

57. The method set forth in claim 55 wherein the step of computing similarities comprises the steps of:

computing fingerprints of the stored copy and the other stored copies and

comparing the fingerprint of the stored copy with the fingerprints of the other stored copies.

58. The method set forth in claim 57 further comprising the step of:

relating a stored copy's fingerprint to the stored copy.

59. The method set forth in claim 55 wherein:

the second identifiers are second document locators specifying stored copies in the repository server

60. The method set forth in claim 59 wherein;

the first identifier is a second document locator specifying a stored copy in the repository server.

61. A data storage device characterized in that:

the data storage device contains code which when executed in a processor implements the method set forth in claim 55.

62. A method of comparing digital documents,
the method comprising the steps of:

making fingerprints of the documents to be compared; and

comparing the fingerprints to determine a similarity value indicating a degree of similarity of the documents being compared, the similarity value being capable of expressing degrees of similarity in addition to than identical or not identical.

63. The method set forth in claim 62 wherein

the digital documents include links to other digital documents and

the method further comprises the step of:

determining from a comparison of the links in the documents to be compared whether the documents are similar enough to justify performance of the step of comparing fingerprints.

64. The method set forth in claim 63 wherein:

in the step of determining from a comparison of the links, the numbers of links in the documents are compared.

65. The method set forth in claim 63 wherein:

in the step of determining from a comparison of the links, the destinations of the links in the documents are compared.

66. The method set forth in claim 62 wherein:

the digital documents are identified by identifiers; and

the method includes the step of relating the fingerprint for a digital document to the digital document's identifier,
whereby digital documents whose identifiers are known and that have fingerprints may be compared without reference to the documents or copies thereof.

67. The method set forth in claim 66 wherein:

the identifiers are document locators used to locate the documents in a network.

68. The method set forth in claim 62 further comprising the step of:

relating a given one of the digital documents to similarity values for a set of other ones of the digital documents.

69. The method set forth in claim 68 wherein:

the digital documents are identified by identifiers; and

in the step of relating, the identifier for the given one of the digital documents is related to identifiers and similarity values for the set of other ones of the digital documents.

70. The method set forth in claim 69 wherein:

the identifiers are document locators used to locate the documents in a network.

71. A data storage device characterized in that:

the data storage device contains code which when executed in a processor implements the method set forth in claim 62.

72. A method of making a fingerprint for a digital document that contains both structural information and content information,

the method comprising the steps of:

encoding the structural information using a first encoding that preserves semantic information about the structural information; and

encoding the content information using a second encoding.

73. The method set forth in claim 72 wherein:

in the step of encoding the structural information, the structural information is encoded in an order in which the structural information occurs in the digital document; and

content information is encoded as it is encountered while encoding the structural information.

74. The method set forth in claim 73 wherein:

~~the structural information is HTML tags.~~

75. A data storage device characterized in that:

the data storage device contains code which when executed in a processor implements the method set forth in claim 72.

76. A method of comparing a first fingerprint for a first digital document with a second fingerprint for a second digital document, the fingerprints containing both structural and content information from the digital documents and being made by encoding the structural information using a first encoding that preserves semantic information about the structural information and encoding the content information using a second encoding and

the method comprising the steps of:

1. finding an encoding of structural information in the first fingerprint;
2. finding a substring in the second fingerprint that matches a substring in the first fingerprint that begins at the found encoding;
3. adding the length of the found substring to a running length total;
4. finding another encoding of structural information in the first fingerprint that is not contained in any found substring and repeating steps 1-3 with the other encoding;
5. repeating steps 1-4 until no further encodings of structural information can be found in step 4; and
6. using the length of one of the fingerprints and the running length total to compute a similarity value.

77. A data storage device characterized in that:

the data storage device contains code which when executed in a processor implements the method set forth in claim 76.

78. A document locator for a repository server in a network, the document locator locating a copy of a document that is stored in the repository server and the copy being a copy of a document that was locatable in the network at the time the copy was made by another document locator,

the document locator comprising:

an identifier for the repository server in the network;

the other document locator; and

an identifier for the stored copy in the server,

whereby the repository server is able to relate the stored copy specified by the identifier to the other document locator.

79. The document locator set forth in claim 78 wherein:

the repository server responds to a received document locator received in the repository server by providing the stored copy identified by the identifier only if the document specified by the other document locator is unavailable in the network.

80. The document locator set forth in claim 78 wherein:

each stored copy in the repository server is related to the other document locator at which the stored copy was locatable when the stored copy was stored; and

the repository server responds to the received document locator by providing a list of document locators for other stored copies that are related to the other document locator in the received document locator.

81. The document locator set forth in claim 78 wherein:

the document locator is a universal resource locator wherein the host name specifies the repository server and parameters include a first parameter whose value is the other document locator, a second parameter whose value is the identifier for the stored copy, and a third parameter for a program that when executed in the repository server provides the stored copy identified by the identifier only if the document specified by the other document locator is unavailable in the network.

82. A data storage device characterized in that:

the data storage device contains code which when executed in a processor constructs the document locator set forth in claim 78.

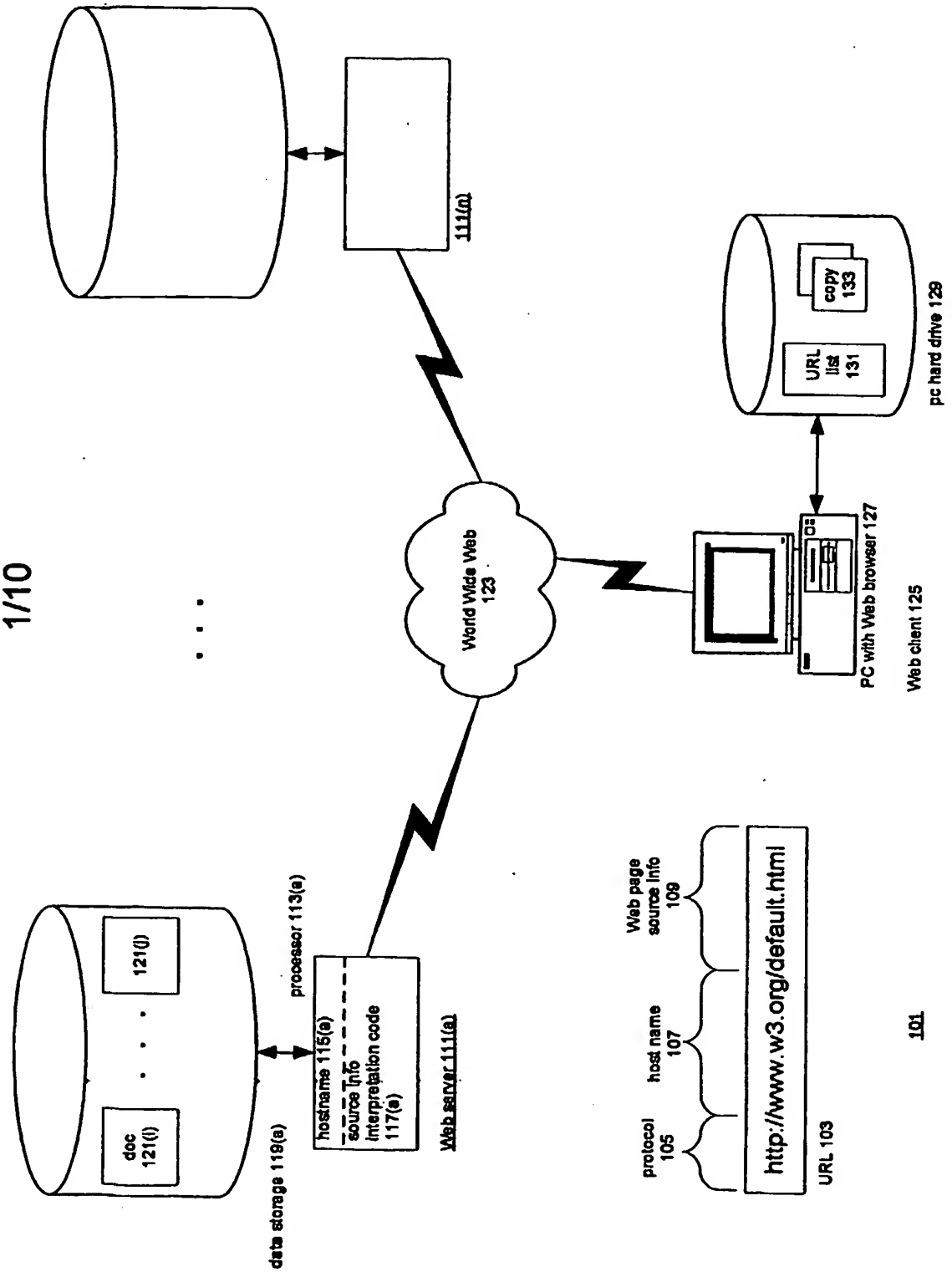


Fig. 1 Prior Art

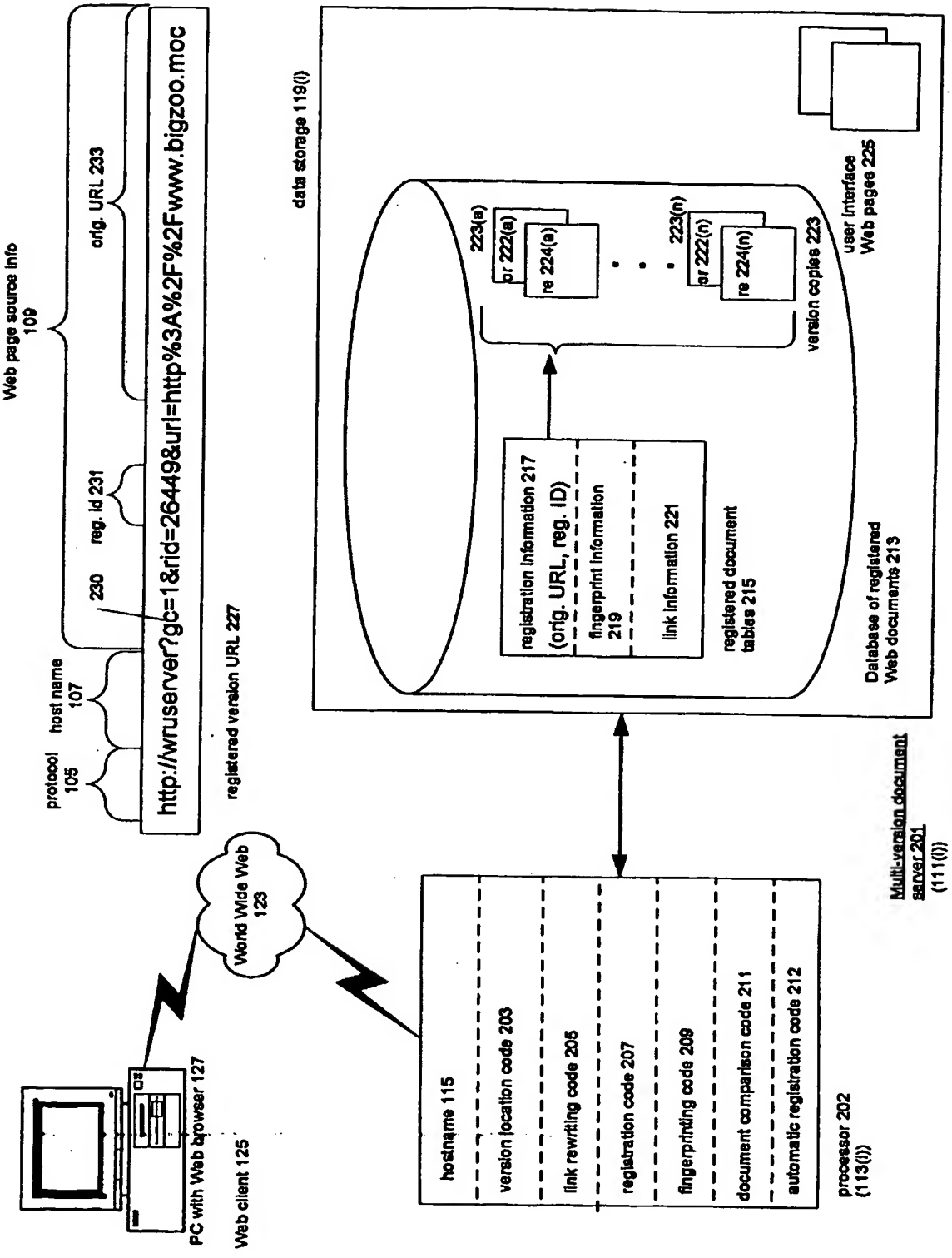

Web page source info 109
orig. URL 233
reg. id 231
230
host name 107
protocol 105
http://wruserver?gc=1&rid=26449&url=http%3A%2F%2Fwww.bigzoo.moc
registered version URL 227
data storage 119(i)
223(a)
cr 222(a)
re 224(a)
223(n)
cr 222(n)
re 224(n)
version copies 223
user interface Web pages 225
registration information 217 (orig. URL, reg. ID)
fingerprint information 219
link information 221
registered document tables 215
Database of registered Web documents 213
Multi-version document server 201 (111(i))
World Wide Web 123
PC with Web browser 127
Web client 125
hostname 115
version location code 203
link rewriting code 205
registration code 207
fingerprinting code 209
document comparison code 211
automatic registration code 212
processor 202 (113(i))

Fig. 2

3/10

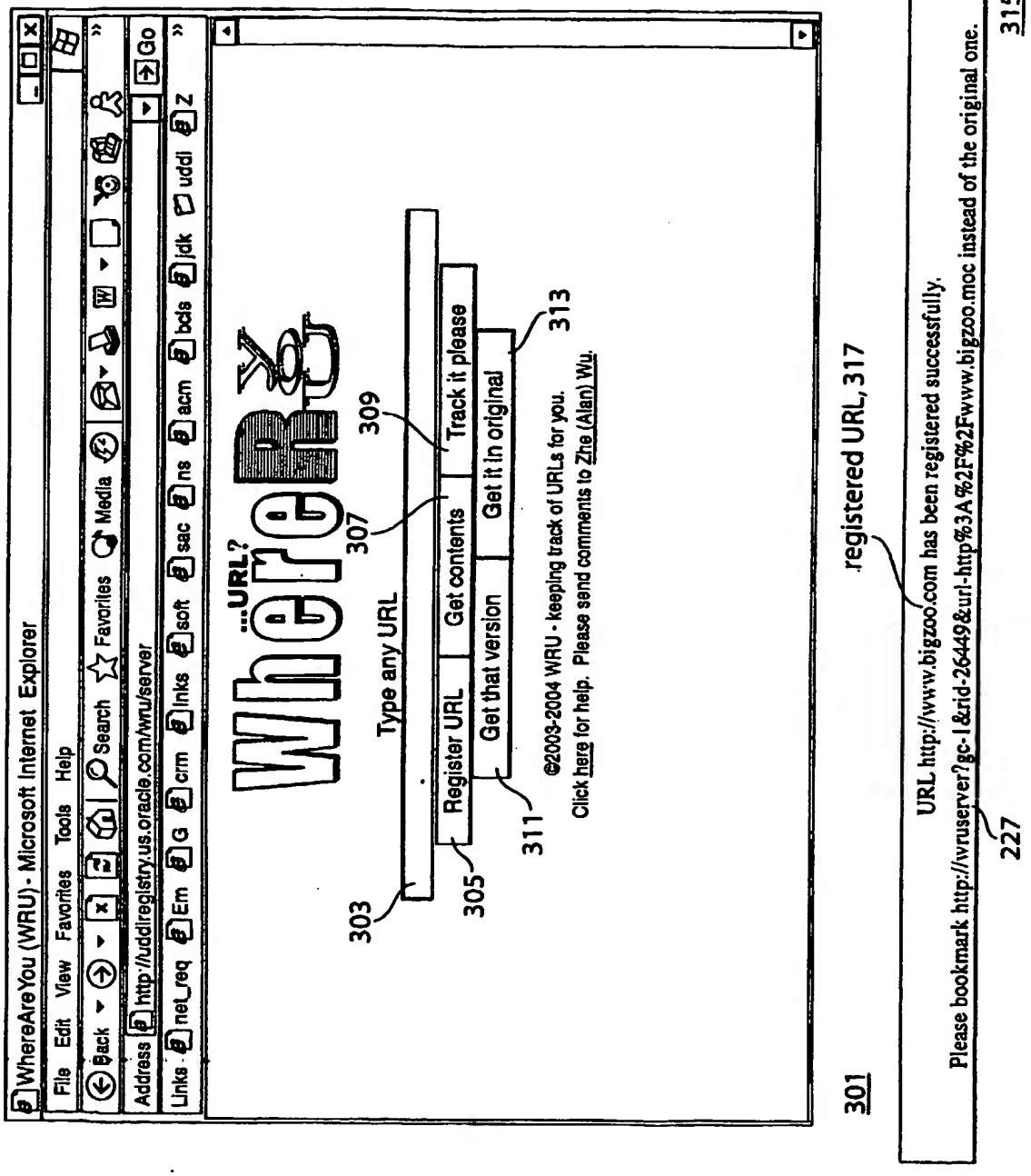
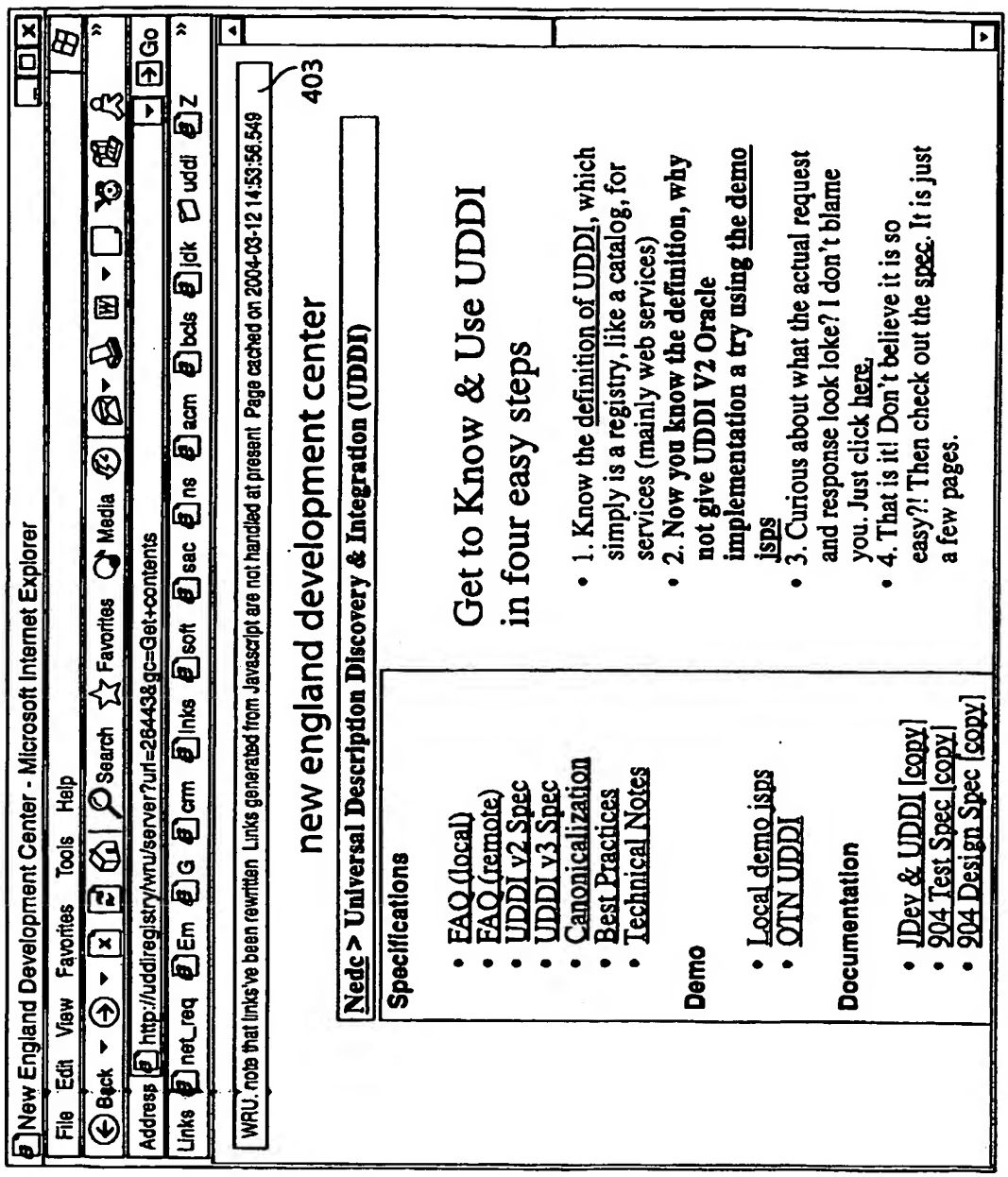

WhereAreYou (WRU) - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://uddiregistry.us.oracle.com/wru/server
Links net_req Em G crm links soft sac ns acm bcls jdk uddi Z
Go
...URL?
WhereRU
Type any URL
Register URL
Get contents
Track it please
Get that version
Get it in original
©2003-2004 WRU - keeping track of URLs for you.
Click here for help. Please send comments to Zhe (Alan) Wu.
303
305
307
309
311
313
301
registered URL, 317
URL http://www.bigzoo.com has been registered successfully.
Please bookmark http://wruserver?gc-1&rid-26449&url-http%3A%2F%2Fwww.bigzoo.moc instead of the original one.
227
315

Fig. 3


New England Development Center - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://uddiregistry/wru/server?url=26443&gc=Get+contents
Go
Links net_req Em G crm links soft sac ns acm bds jdk udd
WRU: note that links've been rewritten Links generated from Javascript are not handled at present Page cached on 2004-03-12 14:53:56.549
403
new england development center
Nedc > Universal Description Discovery & Integration (UDDI)
Specifications
FAQ (local)
FAQ (remote)
UDDI v2 Spec
UDDI v3 Spec
Canonicalization
Best Practices
Technical Notes
Demo
Local demo isps
OTN UDDI
Documentation
JDev & UDDI [copy]
204 Test Spec [copy]
204 Design Spec [copy]
Get to Know & Use UDDI
in four easy steps
1. Know the definition of UDDI, which simply is a registry, like a catalog, for services (mainly web services)
2. Now you know the definition, why not give UDDI V2 Oracle implementation a try using the demo isps
3. Curious about what the actual request and response look loke? I don't blame you. Just click here.
4. That is it! Don't believe it is so easy?! Then check out the spec. It is just a few pages.

401

Fig. 4

5/10

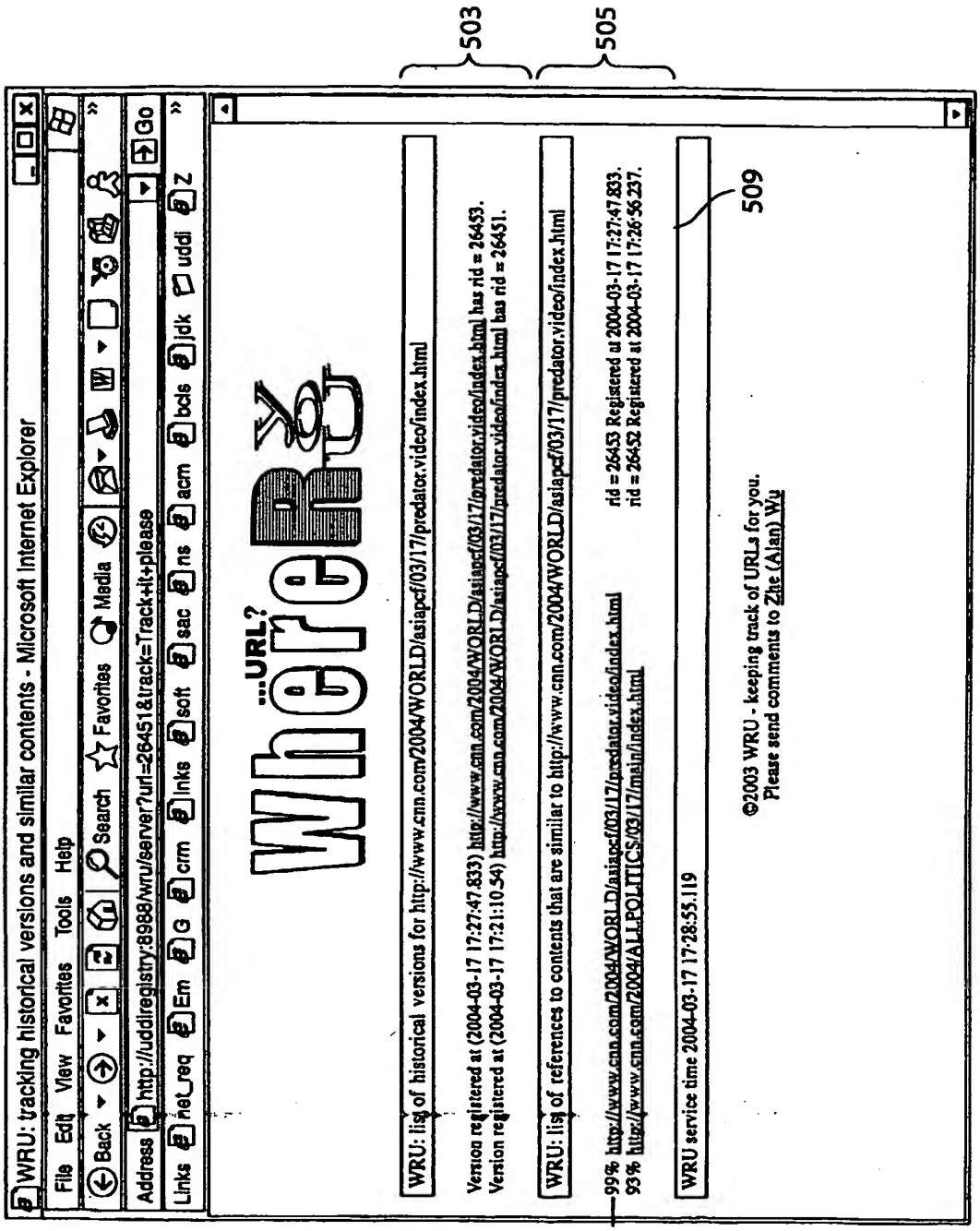

WRU: tracking historical versions and similar contents - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://uddiregistry:8988/wru/server?url=26451&track=Track+it+please
Links net_req Em G crm links soft sac ns acm bcls jdk uddi Z
...URL? Where?
WRU: list of historical versions for http://www.cnn.com/2004/WORLD/asiapcf/03/17/predator.video/index.html
Version registered at (2004-03-17 17:27:47.833) http://www.cnn.com/2004/WORLD/asiapcf/03/17/predator.video/index.html has rid = 26453.
Version registered at (2004-03-17 17:21:10.54) http://www.cnn.com/2004/WORLD/asiapcf/03/17/predator.video/index.html has rid = 26451.
WRU: list of references to contents that are similar to http://www.cnn.com/2004/WORLD/asiapcf/03/17/predator.video/index.html
99% http://www.cnn.com/2004/WORLD/asiapcf/03/17/predator.video/index.html rid = 26453 Registered at 2004-03-17 17:27:47.833.
93% http://www.cnn.com/2004/ALLPOLITICS/03/17/main/index.html rid = 26452 Registered at 2004-03-17 17:26:56.237.
WRU service time 2004-03-17 17:28:55.119
©2003 WRU - keeping track of URLs for you.
Please send comments to Zhe (Alan) Wu
503
505
507
509
501

Fig. 5

6/10

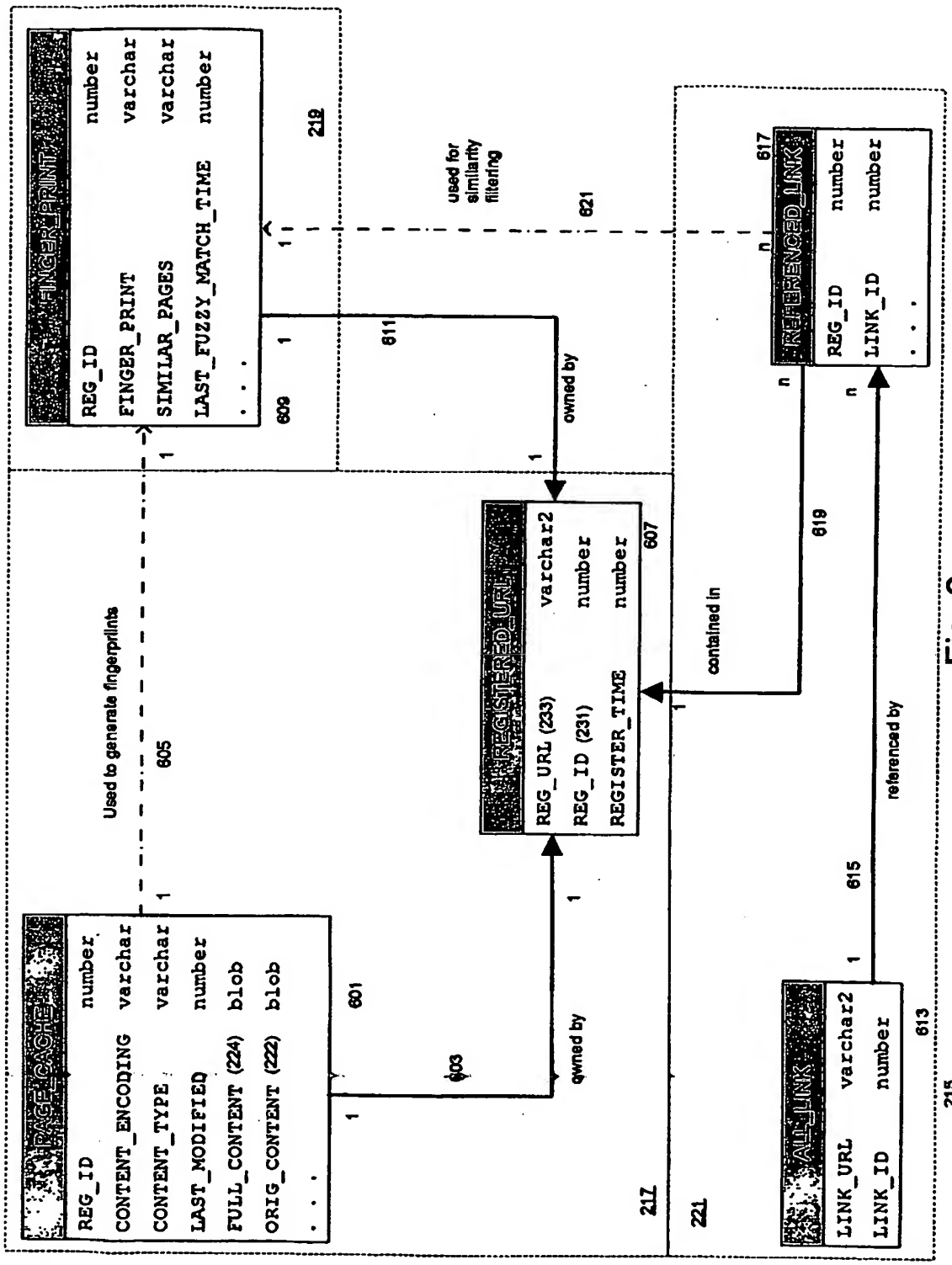
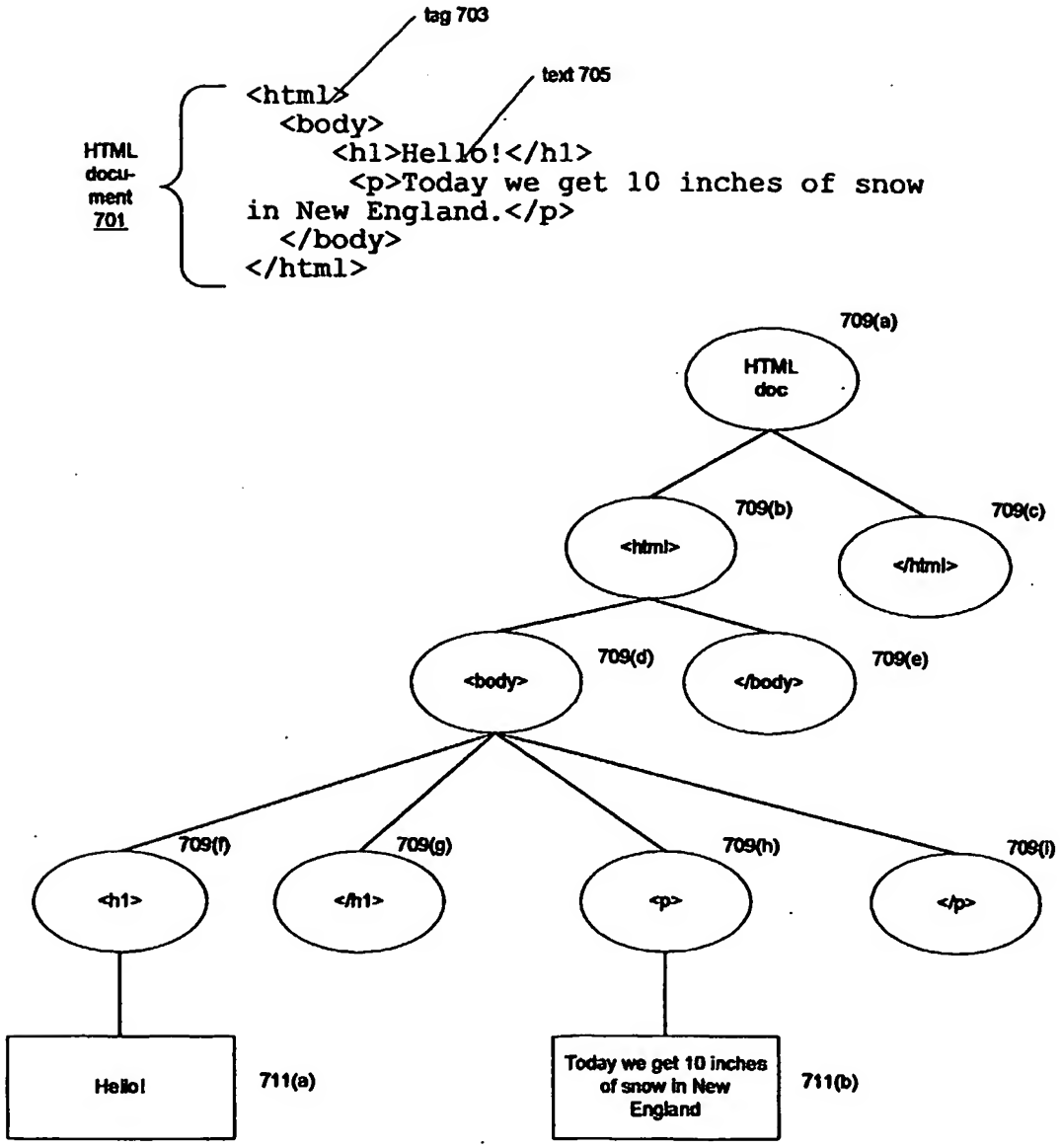

FINGER_PRINT
REG_ID number
FINGER_PRINT varchar
SIMILAR_PAGES varchar
LAST_FUZZY_MATCH_TIME number
. . .
609
219
used for similarity filtering
621
617
REFERENCED_LINK
REG_ID number
LINK_ID number
. . .
owned by
611
Used to generate fingerprints
605
PAGE_CACHE
REG_ID number
CONTENT_ENCODING varchar
CONTENT_TYPE varchar
LAST_MODIFIED number
FULL_CONTENT (224) blob
ORIG_CONTENT (222) blob
. . .
601
603
owned by
REGISTERED_URL
REG_URL (233) varchar2
REG_ID (231) number
REGISTER_TIME number
607
contained in
619
referenced by
615
ALL_LINK
LINK_URL varchar2
LINK_ID number
613
217
221
215

Fig. 6


tag 703
text 705
HTML document 701
<html>
<body>
<h1>Hello!</h1>
Today we get 10 inches of snow in New England.
</body>
</html>
709(a)
HTML doc
709(b)
<html>
709(c)
</html>
<body>
709(d)
</body>
709(e)
709(f)
<h1>
709(g)
</h1>
709(h)

709(i)

Hello!
711(a)
Today we get 10 inches of snow in New England
711(b)
707

Fig. 7

803

HTML start tag	Encoded start tag	HTML end tag	Encoded end tag
<HTML>	<j	</HTML>	'j>
<BODY>	<5	</BODY>	'5>
any other start tag	<X	any other end tag	'X>
<P>	<v	</P>	'v>

HTML encodings 801

```
807 fingerprint     := ""; // empty string to begin with
    while (html-parsing-not-over)
    do
      if (hit-html-start-tag) then
811     fingerprint := fingerprint + map(starting-tag-name);
      end if;

      if (hit-html-ending-tag) then
813     fingerprint := fingerprint + map(ending-tag-name);
809   end if;

      if (hit-html-text) then
        fingerprint := fingerprint + firstTwoChars(text) +
815         toChar(length(text)% 58 + 64);
        // Other alternatives like MD5, hashing are omitted here
      end if;
    end while;

817 return fingerprint; // the final finger print
```

805

821 823 825

<j<5<XheF'X><vTon'v>'5>'j>

fingerprint 819

Fig.8

```
        // Assume two fingerprints are passed in
        // fp1 and fp2 (assume fp1 is not longer than fp2)
        lastPos    := 1;
903     delimPos   := locateSubString(fp1, '<', lastPos);
        similarity := 0;

        while (delimPos > 0)
        do
906       matchLen := length(longestSubStr(fp1, fp2, delimPos));
905       if (matchLen >= 6) then
907         similarity := similarity + matchLen;
          end if;
          lastPos    := delimPos + max(matchLen, 1);
          // Advance to the next starting point.
909       delimPos := locateSubString(fp1, '<', lastPos);
        end while;

        // Get the percentage
911     return similarity / length(fp1);

901
```

Fig. 9

Step 1:
// Assume we have two numbers:
// linkNum1 for number of reference links (URLs) for the requested page
//
// linkNum2 for number of reference links (URLs) for a page
// (Call it candidate page) in the system's repository

The candidate page will pass the filtering *only if*:

$$(linkNum1 + linkNum2 <= 7)$$ 1003

or $$\frac{| linkNum1 - linkNum2 |}{(linkNum1 + 2 + linkNum2)} < 0.3333$$ 1005

1001

Step 2:
// Assume we have two numbers:
// totalLinksNum: the number of reference links (URLs) for the requested page
//
// sameLinksNum: the number of the same links (URLs) for both pages

The candidate page will pass the filtering *only if*:

$$\frac{sameLinksNum + 3}{totalLinksNum + 3} > 0.45$$ 1011 1013 1015

1007

Fig. 10